

Semantic Associations via Regular Relations: Extending Conjunctive Regular Path Queries

Pablo Barceló (U. of Chile)
Carlos Hurtado (UAI, Chile)
Leonid Libkin (U. of Edinburgh)
Peter Wood (U. of London)

A shortcoming of SQL: Recursion

Well-known that SQL does not handle **recursion** over relational databases.

A shortcoming of SQL: Recursion

Well-known that SQL does not handle **recursion** over relational databases.

Adding arbitrary recursion to SQL yields logics with undesirable computational properties (data and combined complexity, query containment, etc).

A shortcoming of SQL: Recursion

Well-known that SQL does not handle **recursion** over relational databases.

Adding arbitrary recursion to SQL yields logics with undesirable computational properties (data and combined complexity, query containment, etc).

This problem has been tackled by adding restricted forms of recursion to SQL:

- ▶ All these additions are rather ad-hoc and based on the idea of extending SQL with well-behaved fragments.
- ▶ **Question:** Are those fragments capable of expressing what we have in mind?

Recursion in graphs

But we probably don't need recursion in every possible domain.

Idea: Restrict recursion to domains where it is needed.

Recursion in graphs

But we probably don't need recursion in every possible domain.

Idea: Restrict recursion to domains where it is needed.

One domain where recursion naturally arises is **graph** databases, i.e. databases that represent links between objects. (Think of semistructured data, for instance).

These databases are considered to be **navigated** (more than queried).

Graph database: Example

Example: The *airline flights* database, such that:

- ▶ Nodes represent cities, and
- ▶ there is an edge between cities u and v labeled a iff there is a **direct** flight from u to v with airline a .

Graph database: Example

Example: The *airline flights* database, such that:

- ▶ Nodes represent cities, and
- ▶ there is an edge between cities u and v labeled a iff there is a **direct** flight from u to v with airline a .

Typical queries one would like to express:

- ▶ Which pairs of cities are connected by a sequence of LAN flights?
- ▶ Which tuples of cities (u, v, w) satisfy that u and v are connected by a sequence of LAN flights and v and w are connected by a sequence of flights in which no LAN flight occurs?

Graph databases

Let Σ be a finite alphabet. A **graph** database over Σ (called Σ -graph) is a tuple $G = (V, E, \lambda)$ such that:

- ▶ (V, E) is a simple and directed graph, and
- ▶ λ is the **labeling** function that assigns a letter in Σ to each edge in E .

A **path** in G from v_1 to v_n is a sequence $\rho = v_1 v_2 \cdots v_n$, such that $(v_i, v_{i+1}) \in E$, for each $1 \leq i < n$.

The **label** of ρ is the following string in Σ^* :

$$\lambda((v_1, v_2))\lambda((v_2, v_3)) \cdots \lambda((v_{n-1}, v_n)).$$

DATALOG over graph databases

Let us consider the simplest query language for relational databases: **Unions of conjunctive queries** (UNION-SELECT-FROM-WHERE fragment of SQL).

DATALOG over graph databases

Let us consider the simplest query language for relational databases: **Unions of conjunctive queries** (UNION-SELECT-FROM-WHERE fragment of SQL).

Consider now its extension with recursion over graph databases: **DATALOG**.

DATALOG over graph databases

Let us consider the simplest query language for relational databases: **Unions of conjunctive queries** (UNION-SELECT-FROM-WHERE fragment of SQL).

Consider now its extension with recursion over graph databases: **DATALOG**.

Then,

	Data complexity	Combined complexity	Query containment
Unions of CQs	AC_0	NP-complete	NP-complete
Datalog	P _{TIME} -complete	EXPTIME-complete	Undecidable

DATALOG over graph databases

Conclusion: As oposed to UCQs, one can hardly implement DATALOG over graph databases, since its data complexity is not parallelizable and query optimization is not computable.

DATALOG over graph databases

Conclusion: As opposed to UCQs, one can hardly implement DATALOG over graph databases, since its data complexity is not parallelizable and query optimization is not computable.

But most of the queries over graph databases that arise in practice are of the simpler form Q :

$$Ans(x, y) \leftarrow (x, \pi, y), R(\pi),$$

where R is a regular expression over Σ .

DATALOG over graph databases

Conclusion: As opposed to UCQs, one can hardly implement DATALOG over graph databases, since its data complexity is not parallelizable and query optimization is not computable.

But most of the queries over graph databases that arise in practice are of the simpler form Q :

$$Ans(x, y) \leftarrow (x, \pi, y), R(\pi),$$

where R is a regular expression over Σ .

Semantics: Given Σ -graph G , the evaluation $Q(G)$ outputs all pairs (a, b) of nodes such that there exists a path between a and b with label in R .

Regular path queries

Queries of the form $Ans(x, y) \leftarrow (x, \pi, y), R(\pi)$ are called **regular path queries (RPQs)**.

For instance, the query

“Which pairs of cities are connected by a sequence of LAN flights?”

can be expressed by the RPQ: $Ans(x, y) \leftarrow (x, \pi, y), LAN^+(\pi)$.

Regular path queries

Queries of the form $Ans(x, y) \leftarrow (x, \pi, y), R(\pi)$ are called **regular path queries (RPQs)**.

For instance, the query

“Which pairs of cities are connected by a sequence of LAN flights?”

can be expressed by the RPQ: $Ans(x, y) \leftarrow (x, \pi, y), LAN^+(\pi)$.

On the other hand, the query $Q_{notsimple}$

“Which tuples of cities (u, v, w) satisfy that u and v are connected by a sequence of LAN flights and v and w are connected by a sequence of flights in which no LAN flight occurs?”

cannot be expressed as a RPQ.

Conjunctive regular path queries

The **conjunctive regular path queries (CRPQs)** are those of the form Q :

$$Ans(\bar{z}) \leftarrow \bigwedge_{1 \leq i \leq m} (x_i, \pi_i, y_i), R_i(\pi_i),$$

where \bar{z} is a tuple of distinct variables in $\bigcup_{1 \leq i \leq m} \{x_i, y_i\}$ and each R_i is a regular expression over Σ .

Conjunctive regular path queries

The **conjunctive regular path queries (CRPQs)** are those of the form Q :

$$Ans(\bar{z}) \leftarrow \bigwedge_{1 \leq i \leq m} (x_i, \pi_i, y_i), R_i(\pi_i),$$

where \bar{z} is a tuple of distinct variables in $\bigcup_{1 \leq i \leq m} \{x_i, y_i\}$ and each R_i is a regular expression over Σ .

Semantics: Given Σ -graph G and valuation σ from the variables in $\bigcup_{1 \leq i \leq m} \{x_i, y_i\}$ into the nodes of G , we say that σ **satisfies** Q wrt G , if for each $i \in [1, m]$ there exists a path from $\sigma(x_i)$ to $\sigma(y_i)$ such that its label is in R_i .

Conjunctive regular path queries

The **conjunctive regular path queries (CRPQs)** are those of the form Q :

$$\text{Ans}(\bar{z}) \leftarrow \bigwedge_{1 \leq i \leq m} (x_i, \pi_i, y_i), R_i(\pi_i),$$

where \bar{z} is a tuple of distinct variables in $\bigcup_{1 \leq i \leq m} \{x_i, y_i\}$ and each R_i is a regular expression over Σ .

Semantics: Given Σ -graph G and valuation σ from the variables in $\bigcup_{1 \leq i \leq m} \{x_i, y_i\}$ into the nodes of G , we say that σ **satisfies** Q wrt G , if for each $i \in [1, m]$ there exists a path from $\sigma(x_i)$ to $\sigma(y_i)$ such that its label is in R_i .

The evaluation $Q(G)$ outputs all those tuples of the form $\sigma(\bar{z})$, for some σ that satisfies Q wrt G .

Observations about CRPQs

First, the query $Q_{notsimple}$ can be expressed as the following CRPQ:

$$Ans(x, y, z) \leftarrow (x, \pi_1, y), LAN^+(\pi_1), \\ (y, \pi_2, z), \overline{(\Sigma^* \cdot LAN \cdot \Sigma^*)}(\pi_2).$$

Observations about CRPQs

First, the query $Q_{notsimple}$ can be expressed as the following CRPQ:

$$Ans(x, y, z) \leftarrow (x, \pi_1, y), LAN^+(\pi_1), \\ (y, \pi_2, z), \overline{(\Sigma^* \cdot LAN \cdot \Sigma^*)}(\pi_2).$$

Second, extending RPQs we allow some of the variables in the body not to be projected in the head.

Observations about CRPQs

First, the query $Q_{notsimple}$ can be expressed as the following CRPQ:

$$Ans(x, y, z) \leftarrow (x, \pi_1, y), LAN^+(\pi_1), \\ (y, \pi_2, z), \overline{(\Sigma^* \cdot LAN \cdot \Sigma^*)}(\pi_2).$$

Second, extending RPQs we allow some of the variables in the body not to be projected in the head.

Third, CRPQs form a subclass of DATALOG [Consens & Mendelzon, '90].

Most studied problems wrt CRPQs

The most studied problems wrt CRPQs are the following:

- ▶ Combined complexity.
- ▶ Data complexity.
- ▶ Query containment.

We go one by one here.

Combined complexity

This is the complexity of the query evaluation problem when both query and data are seen as input:

PROBLEM:	CRPQ evaluation.
INPUT:	A Σ -graph G , a CRPQ Q , and a tuple \bar{t} of nodes in G .
QUESTION:	Does \bar{t} belong to $Q(G)$?

Proposition

(Folklore) The CRPQ evaluation problem is NP-complete.

That is, it is not worse than for the class of CQs. Further, as for the latter, the complexity is improved if queries are *acyclic*.

Combined complexity

Proposition

(Folklore) The CRPQ evaluation problem is NP-complete.

That is, it is not worse than for the class of CQs. Further, as for the latter, the complexity is improved if queries are *acyclic*.

The CRPQ $Ans(\bar{z}) \leftarrow \bigwedge_{1 \leq i \leq m} (x_i, \pi_i, y_i), R_i(\pi_i)$ is **acyclic**, if the undirected graph G_Q defined as follows is also acyclic:

- ▶ The nodes of G_Q are the variables in $\bigcup_{1 \leq i \leq m} \{x_i, y_i\}$, and
- ▶ there is an edge between nodes u and v iff $u = x_i$ and $v = y_i$, for some $1 \leq i \leq m$.

Then,

Combined complexity

Proposition

(Folklore) The CRPQ evaluation problem is NP-complete.

That is, it is not worse than for the class of CQs. Further, as for the latter, the complexity is improved if queries are *acyclic*.

The CRPQ $Ans(\bar{z}) \leftarrow \bigwedge_{1 \leq i \leq m} (x_i, \pi_i, y_i), R_i(\pi_i)$ is **acyclic**, if the undirected graph G_Q defined as follows is also acyclic:

- ▶ The nodes of G_Q are the variables in $\bigcup_{1 \leq i \leq m} \{x_i, y_i\}$, and
- ▶ there is an edge between nodes u and v iff $u = x_i$ and $v = y_i$, for some $1 \leq i \leq m$.

Then,

Proposition

(Folklore) The CRPQ evaluation problem can be solved in PTIME, if restricted to the class of acyclic queries.

But queries are, in this context, much smaller than databases.

It is not completely illogical to consider **queries to be fixed**. This corresponds to the data complexity analysis.

PROBLEM: CRPQ evaluation wrt Q .

INPUT: A Σ -graph G and a tuple \bar{t} of nodes in G .

QUESTION: Does \bar{t} belong to $Q(G)$?

Theorem (Consens & Mendelzon, '90)

CRPQ evaluation wrt Q can be solved in NLOGSPACE.

Worst than for CQs but still parallelizable!

Theorem (Consens & Mendelzon, '90)

CRPQ evaluation wrt Q can be solved in NLOGSPACE.

Worst than for CQs but still parallelizable!

Indeed, an extension of the class of CRPQs precisely captures the class of NLOGSPACE properties over *ordered* graphs.

Simple paths

What if we want to look for *simple* paths connecting nodes?

This makes sense:

“Which pairs of cities are connected by a sequence of flights, such that at least one flight is with LAN, and no city is visited more than once?”

Theorem (Wood, '89)

There exists a CRPQ Q , such that CRPQ evaluation wrt Q and simple paths is NP-complete.

Query containment

The third problem is query containment, that is useful in many static analysis tasks including query optimization, answering queries using views, etc.

The CRPQ Q is **contained** in the CRPQ Q' , if for every Σ -graph G , $Q(G) \subseteq Q'(G)$.

Theorem (Calvanese et al., '00)

The problem of, given two CRPQs Q_1 and Q_2 , checking whether Q_1 is contained in Q_2 , is EXPSPACE-complete.

Bad complexity. Several interesting subclasses, defined in terms of restricted classes of regular languages, have better behavior (Florescu et al., '98; Deutsch & Tannen, '01).

So far, so good. Why do we need to extend CRPQs then?

So far, so good. Why do we need to extend CRPQs then?

In several new applications, it is desirable to compare paths not only for pertinence to some regular relation, but also to compare **among** paths:

- ▶ Social networks.
- ▶ Biological data.

In general, this is needed for finding **semantic associations** between items in huge graph databases.

Semantic associations: Example

The language RDF/S has been recently proposed [Sheth et al., '03] for finding semantic associations in RDF.

There, RDF properties can be declared to be *subproperties* of other properties (denoted $a \leq b$).

Two paths ρ_1 and ρ_2 are declared to be **semantically isomorphic**, if

$$\rho_1 = u_1 u_2 \cdots u_n \quad \text{and} \quad \rho_2 = v_1 v_2 \cdots v_n,$$

for some $n \geq 0$, and for each $1 \leq i \leq n$, $u_i \leq v_i$ or $v_i \leq u_i$.

Semantic associations: Example

The language RDF/S has been recently proposed [Sheth et al., '03] for finding semantic associations in RDF.

There, RDF properties can be declared to be *subproperties* of other properties (denoted $a \leq b$).

Two paths ρ_1 and ρ_2 are declared to be **semantically isomorphic**, if

$$\rho_1 = u_1 u_2 \cdots u_n \quad \text{and} \quad \rho_2 = v_1 v_2 \cdots v_n,$$

for some $n \geq 0$, and for each $1 \leq i \leq n$, $u_i \leq v_i$ or $v_i \leq u_i$.

But, finding whether RDF resources a and b are the origin of two semantically isomorphic paths is not expressible by a CRPQ!

Regular relations

We want a query language that is capable of expressing this kind of comparisons among paths.

In order to do so, we review the notion of **regular relations**.

These are languages of n -tuples of strings over Σ , for some $n > 0$.

Intuitively, an **n -ary regular relation** is a language of n -tuples (w_1, \dots, w_n) of strings that is definable by an automata over alphabet Σ^n .

This automata runs **synchronously** over the strings in the tuple.

Regular relations

But there is an issue: Strings in the tuple may be of different length.

In order to overcome this problem, we **pad** each string in the tuple with a new symbol \perp until all strings are of the same length.

For a tuple $\bar{s} = (s_1, \dots, s_n)$, we denote by $[\bar{s}]$ the padded version of \bar{s} .

Finally, a language L of n -tuples of strings is **regular** iff the set $\{[\bar{s}] \mid \bar{s} \in L\}$ is regular, i.e. accepted by an automaton over $(\Sigma \cup \{\perp\})^n$.

Regular relations: Example

The following are regular relations:

- ▶ Each regular language.
- ▶ The **prefix** relations \prec and \preceq , as well as the equality $=$ relation between strings.
- ▶ The **equal length** relation el that checks whether two strings have the same length.
- ▶ And, in particular, given a partial order between elements of Σ , the relation *iso* that checks whether two strings are semantically equivalent.

On the other hand, some relations are *not* regular. Most notably, the relations of **subsequence** and **concatenation**.

Properties of regular relations

Regular relations have some good computational properties simply because they are given by automata:

- ▶ Closed under Boolean operations and projection.
- ▶ Emptiness decidable in polynomial time.

But we will see later that adding regular relations to CRPQs comes at a cost anyways.

Paths in the output

But there is yet another reason why we want to extend CRPQs: We want queries to output not only nodes **but also paths** in the graph.

This is motivated by growing interest in recent applications that want the actual associations between elements to be returned.

- ▶ Provenance.
- ▶ Biological data.
- ▶ Social networks.
- ▶ RDF (e.g. SPQARQLer).

Extended CRPQs

An **extended CRPQ (ECRPQ)** is a query Q of the form:

$$Ans(\bar{z}, \bar{\chi}) \leftarrow \bigwedge_{1 \leq i \leq m} (x_i, \pi_i, y_i), \bigwedge_{1 \leq j \leq t} R_j(\bar{\pi}^j),$$

such that:

- ▶ \bar{z} is a tuple of distinct variables in $\bigcup_{1 \leq i \leq m} \{x_i, y_i\}$,
- ▶ $\bar{\chi}$ is a tuple of distinct variables in $\{\pi_1, \dots, \pi_m\}$,
- ▶ each R_j is a regular relation of arity $n_j > 0$, and
- ▶ each $\bar{\pi}^j$ is a tuple of arity n_j over $\{\pi_1, \dots, \pi_m\}$.

ECRPQs: Semantics

Let Q be an ECRPQ of the form

$$\text{Ans}(\bar{z}, \bar{\chi}) \leftarrow \bigwedge_{1 \leq i \leq m} (x_i, \pi_i, y_i), \bigwedge_{1 \leq j \leq t} R_j(\bar{\pi}^j).$$

Semantics: Given Σ -graph G , valuation σ from the variables in $\bigcup_{1 \leq i \leq m} \{x_i, y_i\}$ into the nodes of G , and valuation μ from the variables in $\{\pi_1, \dots, \pi_m\}$ to the paths in G , we say that (σ, μ) **satisfies** Q wrt G , if:

- ▶ For each $i \in [1, m]$, the path $\mu(\pi_i)$ goes from $\sigma(x_i)$ to $\sigma(y_i)$, and
- ▶ for each $j \in [1, t]$, if $\bar{\pi}^j = (\pi_1^j, \dots, \pi_{n_j}^j)$ then $[(\lambda(\mu(\pi_1^j)), \dots, \lambda(\mu(\pi_{n_j}^j)))]$ belongs to R_j .

The evaluation $Q(G)$ outputs all those tuples of the form $(\sigma(\bar{z}), \mu(\bar{\chi}))$, for some (σ, μ) that satisfies Q wrt G .

ECRPQs: Expressive power

Clearly, we can now express the query:

“Check whether RDF resources a and b are the origin of two semantically isomorphic paths”

by the query:

$$Ans() \leftarrow (a, \pi_1, x), (b, \pi_2, y), iso(\pi_1, \pi_2).$$

ECRPQs: Expressive power

Clearly, we can now express the query:

“Check whether RDF resources a and b are the origin of two semantically isomorphic paths”

by the query:

$$Ans() \leftarrow (a, \pi_1, x), (b, \pi_2, y), iso(\pi_1, \pi_2).$$

More in general, we can prove the following:

Proposition

The class of ECRQPs is strictly more expressive than the class of CRPQs.

ECRPQ: Combined complexity

The first problem we study wrt ECRPQs is combined complexity.
Formally,

PROBLEM:	ECRPQ evaluation.
INPUT:	A Σ -graph G , an ECRPQ Q , a tuple \bar{t} of nodes in G , and a tuple $\bar{\rho}$ of paths in G .
QUESTION:	Does $(\bar{t}, \bar{\rho})$ belong to $Q(G)$?

ECRPQ: Combined complexity

The combined complexity of ECRPQs is considerably higher than for CRPQs, since we can codify into it the problem of intersection of REs:

Theorem

The problem of ECRPQ evaluation is PSPACE-complete.

ECRPQ: Combined complexity

The combined complexity of ECRPQs is considerably higher than for CRPQs, since we can codify into it the problem of intersection of REs:

Theorem

The problem of ECRPQ evaluation is PSPACE-complete.

As opposed to CRPQs, this bad behavior is the same even for acyclic queries.

Corollary

The problem of ECRPQ evaluation is PSPACE-complete, even for the class of Boolean and acyclic ECRPQs that do not make use of regular relations of arity > 2 .

Again, it is useful to understand the data complexity of the problem. Formally,

PROBLEM:	ECRPQ evaluation wrt Q .
INPUT:	A Σ -graph G , a tuple \bar{t} of nodes in G , and a tuple $\bar{\rho}$ of paths in G .
QUESTION:	Does $(\bar{t}, \bar{\rho})$ belong to $Q(G)$?

Data complexity

The data complexity of the query evaluation problem is not increased by the addition of regular relations in the body of queries:

Theorem

Let Q be an ECRPQ. The problem of ECRPQ evaluation wrt Q can be solved in NLOGSPACE.

Proof idea: Given ECRPQ Q ,

$$Ans(\bar{z}, \bar{x}) \leftarrow \bigwedge_{1 \leq i \leq m} (x_i, \pi_i, y_i), \bigwedge_{1 \leq j \leq t} R_j(\bar{\pi}^j),$$

construct a unique m -ary regular relation that represents Q and evaluate it over (something like) G^m .

Compact representations

Of course, the number of paths in the output may be infinite, even for a fixed query.

A nice thing about the regular relations, is that its closure properties allows us to give a **compact** representation of the output in terms of an automaton:

Proposition

Let Q be a fixed ECRPQ. Given a graph G , it is possible to construct in polynomial time an automaton \mathcal{A} that “represents” $Q(G)$.

Query containment

The last problem is query containment.

Again, the ECRPQ Q is **contained** in the ECRPQ Q' , if for every Σ -graph G , $Q(G) \subseteq Q'(G)$.

We strongly believe the query containment problem to be decidable.

Unfortunately, proving the query containment problem to be decidable would also resolve the decidability of a long-standing open problem in the area of word combinatorics.

Query containment

The last problem is query containment.

Again, the ECRPQ Q is **contained** in the ECRPQ Q' , if for every Σ -graph G , $Q(G) \subseteq Q'(G)$.

We strongly believe the query containment problem to be decidable.

Unfortunately, proving the query containment problem to be decidable would also resolve the decidability of a long-standing open problem in the area of word combinatorics.

At least it is possible to prove decidability in a restricted case:

Theorem

The problem of, given an ECRPQ Q and a CRPQ Q' , checking whether Q is contained in Q' , is in EXPSPACE.

Extending CRPQs with negation

Another interesting way of extending CRPQs is allowing the use of negation in queries.

This allows to express queries of the form:

“ Find all pairs (a, b) of elements such that no path from a to b is labeled in R . ”

Intuitively, this should be expressed by the query:

$$\neg \exists \pi ((x, \pi, y) \wedge R(\pi)).$$

CRPQs with negation: CRPQ^\neg

Formally, the extension of CRPQ with negation, CRPQ^\neg , is defined by the following grammar:

$$\begin{aligned} \textit{Atom} &:= x = y \mid \pi_1 = \pi_2 \mid (x, \pi, y) \mid R(\pi) \\ \phi, \phi' &:= \textit{Atom} \mid \neg\phi \mid \phi \wedge \phi' \mid \exists x\phi \mid \exists \pi\phi \end{aligned}$$

The semantics is standard, and thus, omitted.

Results about CRPQ^-

About this language:

- ▶ **Combined complexity:** PSPACE-complete.
- ▶ **Data complexity:** NLOGSPACE.
- ▶ **Satisfiability:** Undecidable.

In conclusion, adding negation to queries does not have an effect in terms of data complexity, but it does in terms of combined complexity.

The language ECRPQ^\neg

In order for the study to be complete, we also consider the extension ECRPQ^\neg of CRPQ^\neg with regular relations.

This extension allows atoms of the form $R(\pi_1, \dots, \pi_n)$, where R is an n -ary regular relation over Σ .

The language ECRPQ^\neg

In order for the study to be complete, we also consider the extension ECRPQ^\neg of CRPQ^\neg with regular relations.

This extension allows atoms of the form $R(\pi_1, \dots, \pi_n)$, where R is an n -ary regular relation over Σ .

In this case:

- ▶ **Combined complexity:** Decidable, but nonelementary.
- ▶ **Data complexity:** Can be PSPACE-hard. Best upper bound known is k -fold exponential, where k is the size of the query.

In conclusion, highly unfeasible language.

Conclusions

We have presented extensions of the class of CRPQs with regular relations and negation.

Our research has only unveiled the tip of the iceberg, and many problems remain open:

- ▶ Query containment.
- ▶ Query rewriting using views.
- ▶ Parameterized complexity.
- ▶ Arithmetic comparisons.